A Note on ReLU as Both Function and Gate

A small but useful clarification, stated plainly:

> **A ReLU unit is exactly equivalent to a linear value multiplied by a binary decision.**

This is not an analogy—it is an identity.

You can rewrite ReLU as:

$$\mathrm{ReLU}(x) = x \cdot 1[x>0]$$

So every ReLU unit simultaneously produces:

- a **continuous value** x
- a **binary decision** $1[x>0]$

and the output is simply their product.

Bridging the Two Intuitions

It is natural to see:

- the graph → as "piecewise linear, almost smooth"
- the system → as continuous signal processing

But the identity above shows something precise:

> The apparent smoothness comes from *multiplying by the input*, not from the decision itself.

The **decision is fully discrete**:

- either 0 (off)
- or 1 (on)

The **continuity lives in the value being gated**, not in the gate.

A useful mental separation:

- **Gate**: $1[x>0]$ → discrete, combinatorial

- **Signal**: $x \rightarrow$ continuous

ReLU = **gate × signal**

Why This Matters Structurally

Across a layer:

- Each neuron contributes a binary decision
- These decisions form a **binary vector (activation pattern)**
- That pattern determines which parts of the network are active

So a layer computes:

- a **code** (the gates)
- and **values conditioned on that code**

This is directly analogous to mechanisms already familiar in the field:

- **Mixture of Experts (MoE)**
  → gating network selects experts (often softly)
- **Conditional computation / routing**
  → only part of the model is active per input

The distinction is:

> ReLU implements this **implicitly and at very fine granularity**, using simple threshold decisions instead of an explicit gating module.

From Units to Networks

Once written as “gate × signal,” a network becomes:

- a collection of **binary decisions** (the gates)
- selecting a **linear computation path**

For any fixed input:

- all gates are fixed (0 or 1)
- the network reduces to a **single linear map**

Across inputs:

- different gate patterns → different linear maps

So the network is:

> a large family of linear models, indexed by binary gating patterns

---

Local Behavior: Why It Still Feels Smooth

Within a region where no gate changes:

- all 1[x>0] terms are constant
- the network is exactly linear

Crossing a boundary:

- one gate flips
- the linear map changes

So the overall function is:

- continuous
- piecewise linear
- but **composed of discrete switches between linear regimes**

This reconciles the two views:

- **Function view**: continuous, piecewise linear
- **System view**: discrete gating over linear components

Both are true; they describe different aspects of the same identity.

Practical Consequences

Thinking in “gate × signal” terms makes several things explicit:

- **Activation patterns = routing decisions**
- **Weight matrices = shared resources reused across routes**
- **Generalization = nearby inputs share similar gate patterns**
- **Instability = small input changes flip gates → change routes**

A Minimal Operational Check

To make this concrete in practice:

1. Record activation patterns 1[x>0] for a dataset
2. Measure:
    - how often patterns repeat
    - how sensitive they are to perturbations

This directly exposes:

- how much the model relies on shared routes
- where decision boundaries are sharp

Closing Observation

The key point is simple but easy to overlook:

> **ReLU does not blur the distinction between continuous and discrete—it contains both, explicitly.**

- The **discrete part** selects the computation
- The **continuous part** carries the values

Recognizing this equivalence connects standard ReLU networks to:

- gating and routing (as in mixture-of-experts),
- and, more broadly, to systems that combine **decision structure with linear computation**.